Infinite games recognized by 2-stack visibly pushdown automata (Proof Theory, Computation Theory and Related Topics)

AUTHOR(S):

李, 文娟; 沖坂, 祥平; 田中, 一之

数理解析研究所講究録
第1950巻 2015年 121-137

Infinite games recognized by 2-stack visibly pushdown automata

Wenjuan Li, Shohei Okisaka, Kazuyuki Tanaka

Mathematical Institute, Tohoku University, Sendai, 980-8578, Japan

1 Introduction

Two-player infinite games have been extensively studied in Descriptive Set Theory in the past several decades, accompanied by a celebrated result due to Martin that all Gale-Stewart games with Borel winning conditions are determined. In this paper, we are interested in the following problem: if the winning set X is given as $X = L(\mathcal{M})$, where $L(\mathcal{M})$ is the language recognized by some kind of machine $\mathcal{M}$, whether the game $G(X)$ is determined or not, and if one of the two players has a winning strategy, whether the strategy is computable or not.

Büchi and Landweber (1969) first studied the Gale-Stewart game $G(X)$, where X is an ω-regular language accepted by a Büchi automaton. They showed that one can effectively decide the winner of $G(X)$ and the winning strategy can be constructed by a finite transducer. Thomas (1995) asked whether it can be extended to deterministic context-free ω-languages, which are accepted by Muller pushdown automata. Walukiewicz (2001) presented a positive answer by showing that one can effectively construct winning strategies in parity games played on pushdown graphs. This result implies that the Gale-Stewart games over deterministic context-free ω-languages are determined with computable winning strategies.

Finkel (2001) studied the cases where the winning sets are ω-languages accepted by nondeterministic pushdown automata and showed that in such games, it is undecidable to determine which player has a winning strategy. Finkel (2013) also proved that the determinacy over the class of context-free ω-languages is equivalent to the determinacy over the effective class of analytic sets. Research also extends to infinite games related with some variants of pushdown automata. Well-known variants of pushdown automata include, extensions on the stacks, such as higher-order stacks, multi-stacks, and restrictions on the input alphabet, such as visibly input.

As far as we know, two-player infinite games recognized by 2-stack visibly pushdown automata have not been studied up to now. In this paper, we mainly treat 2-visibly pushdown automata with Boolean combination of Σ^0_1 accepting condition. The input alphabet of a 2-stack visibly pushdown automaton is partitioned into push, pop alphabet for each stack separately, and internal alphabet, which decide its visible actions. We show that there exists an infinite game recognized by 2-stack visibly pushdown automata in which Player II has a winning strategy but no computable one.

2 Preliminaries

A finite word, over an alphabet Σ, is a finite sequence of letters $x = a_1a_2...a_n$, where $a_i \in \Sigma$ for all $i \leq n$. For $i \leq n$, $x(i)$ denotes the ith letter of x. For $i \leq n$, $x[i] = x(1)x(2)...x(i)$, which denotes the initial ith segment of x. ε is the *empty sequence*. $\Sigma^{<\omega}$ denotes the *set of finite words*

over Σ. L is a subset of $\Sigma^{<\omega}$ which is called *a finite language* over Σ. By *u.v or uv*, we denote the concatenation of finite words u and v. For $V \subseteq \Sigma^{<\omega}$, $V^{<\omega} = \{v_1...v_n | n \in \mathbb{N} \text{ and } \forall i \leq n, v_i \in V\}$.

An *infinite word* over Σ is an infinite sequence $a_1a_2...a_n...$, where for all $i \geq 1$, $a_i \in \Sigma$. When σ is an infinite word over Σ, we write $\sigma = \sigma(1)\sigma(2)...\sigma(n)...$ and $\sigma[n] = \sigma(1)\sigma(2)...\sigma(n)$, which is the prefix of σ with length n. Σ^ω denotes the *set of infinite words* over Σ. An *ω-language over* Σ is a subset of Σ^ω. The concatenation of a finite word u and an infinite word v is also written as uv.

Let Σ be a finite set. The topology on Σ^ω is defined by the metric d:

$$d(\alpha, \beta) = 2^{-\ell} \text{ where } \ell = \min\{i | \alpha(i) \neq \beta(i)\}$$

with conventions that $\min \emptyset = +\infty$ and $2^{-\infty} = 0$. Then we say a subset of Σ^ω is open if and only if it is in the form $X\Sigma^\omega = \{uv \in \Sigma^\omega | u \in X \text{ and } v \in \Sigma^\omega\}$, where $X \subset \Sigma^{<\omega}$.

2.1 Pushdown automata

2.1.1 Pushdown automata on finite words

Roughly speaking, a pushdown automaton is a finite automaton equipped with a pushdown stack as shown in Figure 1, which can push and pop the top letter of the stack.

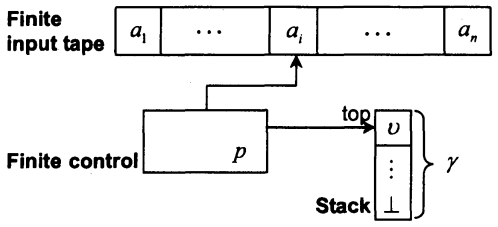

Figure 1 A pushdown automaton with finite input.

This kind of automata can accept the language $\{a^n b^n | n \geq 1\}$. It can store a in the stack when reading some a on the tape. When the letter b is encountered, a top a from the stack can be removed. If the stack becomes empty on the completion of processing a given input, then the pushdown automaton accepts the input. In contrast, this language can not be recognized by finite automata. A finite automaton has only a finite number of states, it cannot remember the number of a's in $a^n b^n$ where n is larger than the number of states of a finite automaton.

The formal definition of pushdown automata is given as follows.

Definition 1. A *(nondeterministic) pushdown automaton* (PDA) is a tuple

$$\mathcal{M} = (Q, \Sigma, \Gamma, q_{\text{in}}, \delta, F)$$

where

- Q is a finite set of states,
- Σ is a finite input alphabet,
- Γ is a finite stack alphabet, which includes a special bottom letter $\perp$,
- $q_{\text{in}} \in Q$ is the initial state,
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma \cup \{\varepsilon\})$ is the transition relation, and

- $F \subseteq Q$ is a set of final states.

The content of stack is denoted by $\gamma \in (\Gamma / \{\perp\})^{<\omega}\perp$. The leftmost letter will be assumed to be on the top of stack, also the bottom letter $\perp$ can never be deleted and the rightmost letter is always $\perp$.

Definition 2. A *configuration* of pushdown automaton is a pair (q, γ), where $q \in Q$ and $\gamma \in (\Gamma / \{\perp\})^{<\omega}\perp$.

For $a \in \Sigma \cup \{\varepsilon\}$, $\gamma \in (\Gamma / \{\perp\})^{<\omega}\perp$, $p, q \in Q$ and $v, \beta \in \Gamma \cup \{\varepsilon\}$, if $(q, \beta) \in \delta(p, a, v)$, then we denote

$$a : (p, v\gamma) \mapsto_{\mathcal{M}} (q, \beta\gamma).$$

$\mapsto_{\mathcal{M}}^{<\omega}$ is the transitive and reflexive closure of $\mapsto_{\mathcal{M}}$.

Definition 3. Let $u = a_1 a_2 ... a_n$ be a finite word over Σ. A finite sequence of configurations $r = (q_i, \gamma_i)_{1 \leq i \leq s+1}$ is called a *run* of a pushdown automaton $\mathcal{M}$ on u, starting from the initial configuration $(q_{\text{in}}, \perp)$, if and only if:

(1) $(q_1, r_1) = (q_{\text{in}}, \perp)$, and

(2) for each $1 \leq i \leq s$, there exists $b_i \in \Sigma \cup \{\varepsilon\}$ such that b_i: $(q_i, \gamma_i) \mapsto_{\mathcal{M}} (q_{i+1}, \gamma_{i+1})$ and such that $a_1 a_2 ... a_n = b_1 b_2 ... b_s$.

The language accepted by $\mathcal{M}$ is

$$L(\mathcal{M}) = \{u \in \Sigma^{<\omega} | \exists \text{ run } r \text{ of } \mathcal{M} \text{ on } u \text{ starting from the initial configuration and stopping with a final state in } F\}.$$

A language is called *context-free* if and only if it is accepted by a pushdown automaton. The class of *context-free* languages is denoted as **CFL**.

CFL is closed under union, concatenation, and Kleene star operation. Note that it is not closed under complement or intersection. However, if L_1 is a context-free language and L_2 is a regular language (which is accepted by finite automata) then their intersection $L_1 \cap L_2$ is a context-free language.

A pushdown automaton $\mathcal{M} = (Q, \Sigma, \Gamma, q_{\text{in}}, \delta, F)$ is said to be deterministic if and only if

$$|\delta(q, a, \gamma)| + |\delta(q, \varepsilon, \gamma)| \leq 1,$$

where $q \in Q$, $a \in \Sigma$ and $\gamma \in \Gamma$. By $|X|$, we denote the number of elements in a finite set X. The class of languages accepted by deterministic pushdown automata is denoted as **DCFL**. **DCFL** is closed under complement, but not closed under union, intersection or Kleene star operation.

We say a language is regular if it can be recognized by a finite automaton. The class of regular languages is denoted as **REG**. It is known that for every regular language L, there exists a deterministic automaton $\mathcal{M}$ such that $L(\mathcal{M}) = L$. A deterministic pushdown automaton can simulate a finite automaton and ignores its stack. On the other hand, the deterministic pushdown automata are less powerful than the nondeterministic ones. For example, $\{ww^R | w \in \Sigma^{<\omega}$ *and* w^R *is the reverse form of* $w\}$, is accepted by nondeterministic pushdown automata but not any deterministic ones. Thus, **REG** $\subsetneq$ **DCFL** $\subsetneq$ **CFL**.

2.1.2 Pushdown automata on infinite words

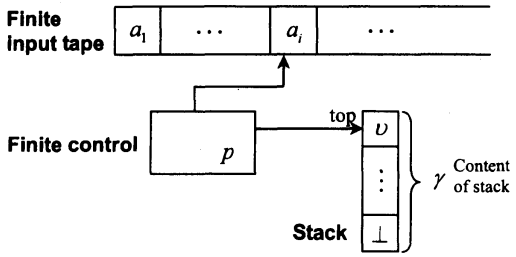

Figure 2 A pushdown automaton with infinite input.

When the input is an infinite sequence of letters as shown in Figure 2, we get an infinite run on such an input.

Definition 4. Let $\sigma = a_1 a_2 ... a_n ...$ be an infinite word over Σ. An infinite sequence of configurations $r = (q_i, \gamma_i)_{i \geq 1}$ is called a *run* of $\mathcal{M}$ on σ, starting from the initial configuration $(q_{\text{in}}, \perp)$, if and only if:

(1) $(q_1, r_1) = (q_{\text{in}}, \perp)$, and

(2) for each $i \geq 1$, there exists $b_i \in \Sigma \cup \{\varepsilon\}$ such that $b_i : (q_i, \gamma_i) \mapsto_{\mathcal{M}} (q_{i+1}, \gamma_{i+1})$ and such that $a_1 a_2 ... a_n ... = b_1 b_2 ... b_n ....$

For every run r, $Inf(r)$ is the set of states that are visited infinitely many times during run r.

Definition 5. A *Büchi pushdown automaton* (BPDA) is a tuple $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_{\text{in}}, F)$ where

- $\mathcal{M}' = (Q, \Sigma, \Gamma, \delta, q_{\text{in}})$ is a pushdown machine which is equipped with no accepting condition, and
- $F \subseteq Q$ is a set of final states.

The *ω-language accepted by $\mathcal{M}$* is

$$L(\mathcal{M}) = \{\sigma \in \Sigma^\omega | \text{there is a run } r \text{ of } \mathcal{M} \text{ on } \sigma \text{ starting from the initial configuration such that } Inf(r) \cap F \neq \emptyset\}.$$

Definition 6. A *Muller pushdown automaton* (MPDA) is a tuple $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_{\text{in}}, \mathcal{F})$ where

- $\mathcal{M}' = (Q, \Sigma, \Gamma, \delta, q_{\text{in}})$ is a pushdown machine, and
- $\mathcal{F} \subseteq \mathcal{P}(Q)$ is a collection of state sets.

The *ω-language accepted by $\mathcal{M}$* is

$$L(\mathcal{M}) = \{\sigma \in \Sigma^\omega | \text{there is a run } r \text{ of } \mathcal{M} \text{ on } \sigma \text{ starting from the initial configuration such that } Inf(r) \in \mathcal{F}\}.$$

Cohen and Gold (1977) gave a characterization theorem for the ω-languages accepted by pushdown automata.

Theorem 1. *Let* **CFL** *be the class of context-free (finite) languages. Then for any ω-language L, the following three conditions are equivalent:*

(1) $L \in \omega\text{-}KC(\mathbf{CFL})$.

(2) There exists a Büchi pushdown automaton that accepts L.

(3) There exists a Muller pushdown automaton that accepts L.

An ω-language is a *context-free ω-language* ($\mathbf{CFL}_\omega$) if and only if it satisfies one of the conditions in the above theorem.

Recall the definition of ω-Kleene closure. For any family $\mathcal{L}$ of finite languages over the alphabet Σ, the *ω-Kleene closure* of $\mathcal{L}$ is $\omega\text{-}KC(\mathcal{L}) = \{\bigcup_{i=1}^{n} U_i V_i^\omega | U_i, V_i \in \mathcal{L}, (1 \leq i \leq n) \text{ and } n \in \mathbb{N}\}$.

Finkel (2001) proved that $\mathbf{CFL}_\omega$ exhausts the finite ranks of Borel hierarchy, i.e., for each $n \geq 1$, there exists some $\mathbf{\Sigma}_n^0$-complete set and some $\mathbf{\Pi}_n^0$-complete set. Finkel (2003) also showed that there exist some context-free ω-languages which are Borel sets of infinite rank. For more detailed and recent studies on context-free ω-languages, there is an survey conducted by Finkel (2014). Löding (2014) provided some new results on decision problems for deterministic pushdown automata on infinite words.

We denote the class of ω-languages accepted by deterministic Muller pushdown automata, the class of deterministic context-free ω-languages ($\mathbf{DCFL}_\omega$). The the class of ω-languages accepted by deterministic Büchi pushdown automaton is strictly included into the class of ω-languages accepted by deterministic Muller pushdown automata. $\mathbf{DCFL}_\omega$ belongs to the Boolean combinations of Σ_2^0.

2.2 Gale-Stewart games

Definition 7. Let $X \subseteq \Sigma^\omega$, where Σ is a finite alphabet. The Gale-Stewart game $G(X)$ is a game of perfect information between two players.

- Player I first chooses $a_1 \in \Sigma$, then Player II chooses $b_1 \in \Sigma$, then Player I chooses $a_2 \in \Sigma$, and so on.
- After ω steps, the two players have produced a word $x = a_1 b_1 a_2 b_2 ...$ of Σ^ω.
- Player I wins the play if and only if $x \in X$, otherwise Player II wins the play.

Definition 8. A *strategy for Player I* is a function

$$f_I : (\Sigma^2)^{<\omega} \to \Sigma$$

from the set of words of even length to Σ. A *strategy for Player II* is a function

$$f_{II} : (\Sigma^2)^{<\omega}\Sigma \to \Sigma$$

from the set of words of odd length to Σ.

Definition 9. Player I *follows* the strategy f_I in a play $a_1 b_1 a_2 b_2 ... a_n b_n ...$ if for each integer $n \geq 1$, $a_n = f_I(a_1 b_1 a_2 a_2 ... a_{n-1} b_{n-1})$. If Player I wins every play in which he has followed the strategy f_I, then we say that the strategy f_I is a *winning strategy* (w.s.) for Player I. The winning strategy for Player II can be defined similarly.

Definition 10.
- Given $X \subseteq \Sigma^\omega$, the Gale-Stewart game $G(X)$ over winning set X, is said to be determined if one of the two players has a winning strategy in $G(X)$.

- For class$\mathcal{C}$ of ω-languages, we denote $\mathbf{Det}(\mathcal{C})$ to assert "Every Gale-Stewart game $G(X)$ is determined, where $X \in \mathcal{C}$".

Table 1 is a summary of some important results on the determinacy of Gale-Stewart games.

Table 1: Summary of classical results on determinacy.

Winning sets	Determinacy results
$\mathbf{\Sigma}_1^0$	Gale and Stewart, 1953
$\mathbf{\Pi}_2^0$	Wolfe, 1955
$\mathbf{\Sigma}_3^0$	Davis, 1964
$\mathbf{\Pi}_4^0$	Paris, 1972
$\mathbf{\Delta}_1^1$ (Borel)	Martin, 1975
Σ_1^1	Martin, 1970 and Harrington, 1978

Besides these classical determinacy results over Borel sets, we are also interested in the determinacy over the winning sets which are recognized by some machines. Table 2 lists the complexity of several languages recognized by various machines, where $\mathcal{B}$ means the Boolean combinition.

Table 2: Complexity of the ω-languages accepted by various machines

Class of ω-languages	Complexity	Remark
$\mathbf{REG}_\omega$	$\subset \mathcal{B}(\Sigma_2^0)$	FA with Büchi conditions
$\cap$		
$\mathbf{DCFL}_\omega$	$\subset \mathcal{B}(\Sigma_2^0)$	determ. PDA with Muller conditions
$\cap$		
$\mathbf{DTM}_\omega$ with Muller	$= \mathcal{B}(\Sigma_2^0)$	determ. Turing machine with Muller conditions
$\mathbf{CFL}_\omega$	$\subset \Sigma_1^1$	PDA with Büchi conditions
$\cap$		
$\mathbf{NTM}_\omega$ with Büchi	$= \Sigma_1^1$	Turing machine with Büchi conditions

In the case that a winning set is recognized by Büchi automaton (i.e., ω-regular languages), Büchi and Landweber (1969) showed that one of the two players has a winning strategy, which can be effectively constructed, which is known as the Büchi-Landweber theorem.

Theorem 2 (Büchi-Landweber, 1969). ***REG**$_\omega$ games are effectively determined with a computable winning strategy.*

This theorem is extended by Walukiewicz (2001) to deterministic Muller pushdown automata.

Theorem 3 (Walukiewicz, 1996). ***DCFL**$_\omega$ games are effectively determined with a computable winning strategy.*

Fridman *et al.* (2011) studied the delay games over deterministic context-free ω-languages. Delay games are kind of infinite games where one player may postpone his choices. It is undecidable to determine a winner in delay games over deterministic context-free ω-languages.

Finkel (2001) proved that it is undecidable to determine the winner in the Gale-Stewart games with winning sets of context-free ω-languages. Recently, Finkel (2013) showed that the determinacy of context-free ω-languages is equivalent to the determinacy of the effective class Σ^1_1.

Theorem 4 (Finkel, 2013). $\mathbf{Det}(\Sigma^1_1) \leftrightarrow \mathbf{Det}(\mathbf{CFL}_\omega)$.

Further extensions are conducted to deterministic higher-order pushdown automata (DHPDA). Higher-order pushdown automata are generalization of pushdown automata, which are equipped with higher-order stacks. Higher-order stacks can be regarded as stacks of stacks structures. The order of higher-order pushdown automata depends on the depth of nested stacks. Higher-order pushdown automata can not only push and pop letter on the top of the stack, but also can push copy of the topmost stack of any order. An example of an order-3 pushdown automaton is illustrated in Figure 3.

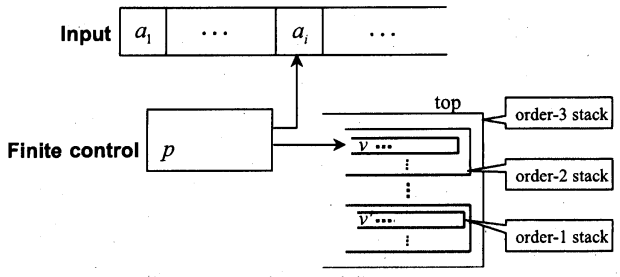

Figure 3 An order-3 pushdown automaton.

The class of ω-languages accepted by higher-order pushdown automata is denoted as $\mathbf{HPDL}_\omega$. By $\mathbf{DHPDL}_\omega$, we denote the ω-languages accepted by deterministic higher-order pushdown automata.

Theorem 5 (Cachat, 2003; Carayol, Hague, Meyer, Ong, 2008). *$\boldsymbol{DHPDL}_\omega$ games are effectively determined with a computable winning strategy.*

A summary of results on determinacy and computable winning strategies in illustrated as follows.

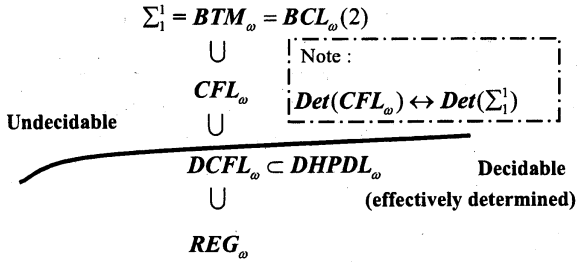

Figure 4 A summary of results on determinacy and computable winning strategy.

3 Infinite game recognized visibly pushdown automata

In this section, we are going to extend the winning set of infinite games to the language recognized by 2-stack visibly pushdown automata. First, we will review the (1-stack) visibly pushdown automata on finite words and infinite words. Then, we consider 2-stack visibly pushdown automata on infinite words, and concentrate on the determinacy and computable winning strategies of infinite games recognized by 2-stack visibly pushdown automata.

3.1 1-stack visibly pushdown automata

Visibly pushdown automata are kind of pushdown automata with restriction on the input alphabet. The *alphabet* is partitioned into **Push**, **Pop**, **Int**. The *transitions* are as follows.

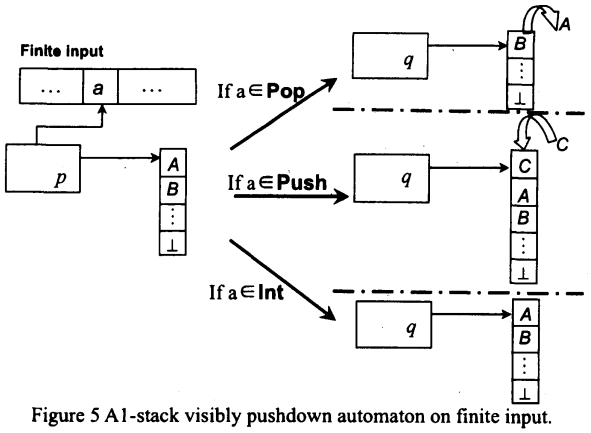

Figure 5 A1-stack visibly pushdown automaton on finite input.

It is an input driven machine. That is, when reading a letter from **Push**, it pushes a letter on the stack, when reading a letter from **Pop**, it pops the top letter from the stack, and when reading a letter from **Int**, it does not touch the stack.

The formal definition is given as follows.

Definition 11. A *visibly pushdown automaton*(VPA) is a tuple $\mathcal{M} = (\Sigma, \Gamma, Q, q_{\text{in}}, \delta, F)$, where

- $\Sigma = \textbf{Push} \cup \textbf{Pop} \cup \textbf{Int}$ is a finite input alphabet,
- Γ is a finite stack alphabet, which contains a special bottom letter $\perp$,
- Q is a finite set of control states,
- $q_{\text{in}} \in Q$ is the initial state,
- $\delta = \delta_{\text{Push}} \cup \delta_{\text{Pop}} \cup \delta_{\text{Int}}$ is a transition relation, where
  - $\star$ $\delta_{\text{Push}} \subseteq Q \times \textbf{Push} \times Q \times (\Gamma / \{\perp\})$,
  - $\star$ $\delta_{\text{Pop}} \subseteq Q \times \textbf{Pop} \times \Gamma \times Q$,
  - $\star$ $\delta_{\text{Int}} \subseteq Q \times \textbf{Int} \times Q$,
- $F \subseteq Q$ is a set of final states.

A visibly pushdown automaton $\mathcal{M}$ is deterministic (DVPA) if the $\delta = \delta_{\text{Push}} \cup \delta_{\text{Pop}} \cup \delta_{\text{Int}}$ is a transition function such that

- $\star$ $\delta_{\text{Push}} : Q \times \textbf{Push} \to Q \times \Gamma / \{\perp\}$,
- $\star$ $\delta_{\text{Pop}} : Q \times \textbf{Pop} \times \Gamma \to Q$,
- $\star$ $\delta_{\text{Int}} : Q \times \textbf{Int} \to Q$,

Definition 12. Let $u = a_1 a_2 ... a_n$ be finite word over Σ. A finite sequence of configurations $r = (q_i, \gamma_i)_{1 \leq i \leq n}$ is called a *run* of a visibly pushdown automaton $\mathcal{M}$ on u, starting from the initial configuration $(q_{\text{in}}, \perp)$, if and only if:

(1) $(q_1, \gamma_1) = (q_{\text{in}}, \perp)$, and

(2) for each $i \in \{1, ..., n\}$,

- $(q_i, a_i, q_{i+1}, \upsilon) \in \delta_{\text{Push}}$ and $\gamma_{i+1} = \upsilon\gamma_i$, or
- $(q_i, a_i, \upsilon, q_i) \in \delta_{\text{Pop}}$ and either $(\upsilon \in \Gamma / \{\perp\}$ and $\gamma_i = \upsilon\gamma_{i+1})$ or $(\upsilon = \perp = \gamma_i = \gamma_{i+1})$, or
- $(q_i, a_i, q_{i+1}) \in \delta_{\text{Int}}$ and $\gamma_i = \gamma_{i+1}$.

A finite word $a_1 ... a_n \in A^{<\omega}$ is recognized by a VPA $\mathcal{M}$ if there exists a run that ends with a state in F. The class of languages recognized by visibly pushdown automata is **VPL**. **VPL** is closed under intersection, union, complementation, concatenation, and Kleene star operation. It is a subclass of **DCFL** and a superclass of **REG** (Alur, Madhusudan, 2004).

Next we will consider 1-stack visibly pushdown automata on infinite input.

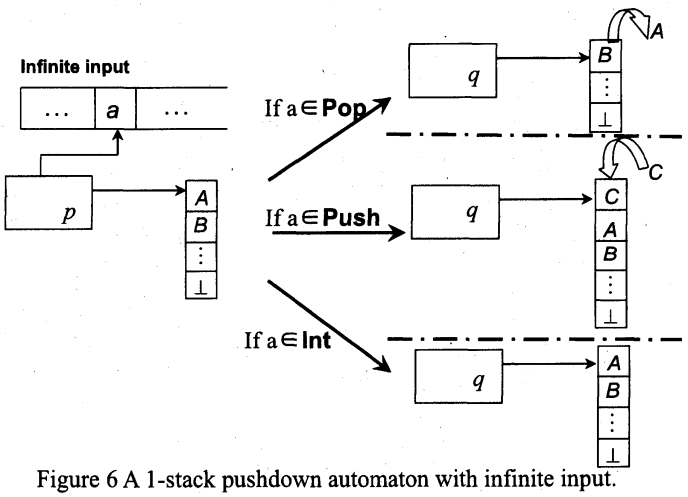

Figure 6 A 1-stack pushdown automaton with infinite input.

Similar with pushdown automata on infinite input, a run of a 1-stack visibly pushdown automaton on $a_1 ... a_n ...$ is an infinite sequence of configurations:

$$(q_0, \perp) \xrightarrow{a_1} (q_1, \gamma_1) ... \xrightarrow{a_n} (q_n, \gamma_n) \xrightarrow{a_{n+1}} ...$$

An infinite word $a_1 ... a_n ... \in A^{\omega}$ is recognized by VPA $\mathcal{M} = (\Sigma.\Gamma, Q, q_{\text{in}}, \delta, F)$ with Büchi accepting condition if there exists a run visiting F infinitely many times. We denote the class of ω-languages accepted by VPA and DVPA as $\mathbf{VPL}_{\omega}$ and $\mathbf{DVPL}_{\omega}$.

Löding, Madhusudan, Serre (2004) showed that $\mathbf{VPL}_{\omega}$ is contained in $\mathcal{B}(\Sigma_3^0)$ sets and $\mathbf{DVPL}_{\omega}$ in $\mathcal{B}(\Sigma_2^0)$. They also proved that visibly pushdown games are determined, where visibly pushdown games are kind of games on graphs generated by visibly pushdown processes. Since $\mathbf{DVPL}_{\omega}$ is a subclass of $\mathbf{DCFL}_{\omega}$, we know that

Theorem 6. *$\mathbf{DVPL}_{\omega}$ games are effectively determined with a computable winning strategy.*

Then the above results on determinacy and computable winning strategies have been extended as follows.

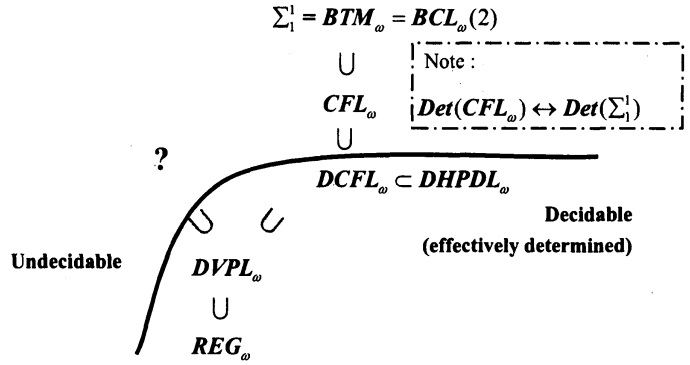

Figure 7 A summary of results on determinacy and computable winning strategy.

On possible further investigation is to look at the 2-stack visibly pushdown automata (2VPA). In the following section, we would like to review 2VPA and 2DVPA (deterministic 2VPA) on finite words, then extend to an ω-languages. Finally, we will focus on the infinite games with winning set recognized by 2DVPA.

3.2 2-stack visibly pushdown automata

For a 2-stack visibly pushdown automaton, the *alphabet* is partitioned into $\mathbf{Push}_1$, $\mathbf{Pop}_1$, $\mathbf{Push}_2$, $\mathbf{Pop}_2$, $\mathbf{Int}$.

Definition 13. A *2-stack visibly pushdown automaton* is a tuple $\mathcal{M} = (\Sigma, \Gamma, Q, q_{\text{in}}, \delta, F)$, where

- $\Sigma = \mathbf{Push}_1 \cup \mathbf{Pop}_1 \cup \mathbf{Push}_2 \cup \mathbf{Pop}_2 \cup \mathbf{Int}$ is a finite input alphabet,
- Γ is a finite stack alphabet, which contains a special bottom letter $\perp$,
- Q is a finite set of states,
- $q_{\text{in}} \in Q$ is the initial state,
- $\delta = \delta_{\text{Push}_1} \cup \delta_{\text{Pop}_1} \cup \delta_{\text{Push}_2} \cup \delta_{\text{Pop}_2} \cup \delta_{\text{Int}}$ is a transition relation, where
  - $\star$ $\delta_{\text{Push}_1} \subseteq Q \times \mathbf{Push}_1 \times Q \times (\Gamma / \{\perp\})$,
  - $\star$ $\delta_{\text{Pop}_1} \subseteq Q \times \mathbf{Pop}_1 \times \Gamma \times Q$,
  - $\star$ $\delta_{\text{Push}_2} \subseteq Q \times \mathbf{Push}_2 \times Q \times (\Gamma / \{\perp\})$,
  - $\star$ $\delta_{\text{Pop}_2} \subseteq Q \times \mathbf{Pop}_2 \times \Gamma \times Q$,
  - $\star$ $\delta_{\text{Int}} \subseteq Q \times \mathbf{Int} \times Q$,
- $F \subseteq Q$ is a set of final states.

A configuration of a 2-stack visible pushdown automaton is in the form (q, γ^1, γ^2), where $q \in Q$, γ^1, γ^2 represent contents of the two stacks and $\gamma^0, \gamma^1 \in (\Gamma / \{\perp\})^{<\omega}\perp$.

Definition 14. Let $u = a_1a_2...a_n$ be finite word over Σ. A finite sequence of configurations $r = (q_i, \gamma_i^1, \gamma_i^2)_{1\leq i\leq n}$ is called a *run* of a 2VPA $\mathcal{M}$ on u, starting from the initial configuration $(q_{\text{in}}, \bot, \bot)$, if and only if:

(1) $(q_1, \gamma_1^1, \gamma_1^2) = (q_{\text{in}}, \bot, \bot)$, and

(2) for each $i \in \{1, ..., n\}$,

- $(q_i, a_i, q_{i+1}, \upsilon) \in \delta_{\text{Push}_1}$ and $\gamma_{i+1}^1 = \upsilon\gamma_i^1$, or
- $(q_i, a_i, \upsilon, q_{i+1}) \in \delta_{\text{Pop}_1}$ and either ($\upsilon \in \Gamma/\ \{\bot\}$ and $\gamma_i^1 = \upsilon\gamma_{i+1}^1$) or ($\upsilon = \bot = \gamma_i^1 = \gamma_{i+1}^1$), or
- $(q_i, a_i, q_{i+1}, \upsilon) \in \delta_{\text{Push}_2}$ and $\gamma_{i+1}^2 = \upsilon\gamma_i^2$, or
- $(q_i, a_i, \upsilon, q_{i+1}) \in \delta_{\text{Pop}_2}$ and either ($\upsilon \in \Gamma/\ \{\bot\}$ and $\gamma_i^2 = \upsilon\gamma_{i+1}^2$) or ($\upsilon = \bot = \gamma_i^2 = \gamma_{i+1}^2$), or
- $(q_i, a_i, q_{i+1}) \in \delta_{\text{Int}}$, $\gamma_i^1 = \gamma_{i+1}^1$, and $\gamma_i^2 = \gamma_{i+1}^2$.

Example 1. *Given* $\Sigma = (\{a\}, \{\overline{a}\}, \{b\}, \{\overline{b}\}, \emptyset)$, *the language*

$$\{(ab)^n\overline{a}^n\overline{b}^n |\ n \in \mathbb{N}\}$$

is recognized by a deterministic 2-stack visibly pushdown automaton (2DVPA).

Example 2. *Given* $\Sigma = (\{a\}, \{c, d\}, \{b\}, \{x, y\}, \emptyset)$, *the language*

$$\{(ab)^n c^i d^{n-i} x^i y^{n-i} |\ n, i \in \mathbb{N} \text{ and } i \leq n\}$$

is recognized by a non-deterministic 2-stack visibly pushdown automaton (2VPA), but not 2DVPA.

Details: The input alphabet is partitioned into $\mathbf{Push}_1 = \{a\}$, $\mathbf{Pop}_1\{c, d\}$, $\mathbf{Push}_2 = \{b\}$, $\mathbf{Pop}_2 = \{x, y\}$, $\mathbf{Int} = \emptyset$.

For a non-deterministic 2VPA,

- while reading a and b, it can push $\sharp$ onto Stack 1 and Stack 2 separately, and
- nondeterministically switch to push $\diamondsuit$ onto both stacks.

Intuitively, this switch corresponds to the guess of what i is. While popping, it checks that $c's$ are used to pop $\diamondsuit$ and $d's$ for $\sharp$, and $y's$ for $\diamondsuit$ and $x's$ for $\sharp$.

The class of finite languages recognized by 2VPA is denoted as **2VPL**. **2VPL** is closed under union, intersection and complement (Carotenuto, Murano, Peron, 2007).

3.3 Main results

There are many ways to extend machines on finite words to infinite words by different accepting conditions as we introduced in the previous sections, such as Büchi condition, Muller condition. In this study, we mainly use a $\mathcal{B}(\Sigma_1^0)$ accepting condition for a 2DVPA $\mathcal{M}$. By $\mathbf{2DVPL}_\omega$, we denote the class of ω-languages accepted by such 2DVPA.

The statement of our main theorem is as follows.

Theorem 7. *There exists a deterministic 2-stack visibly pushdown automaton* $\mathcal{M}$ *such that*

(1) the ω-language $L(\mathcal{M})$ is defined by a Boolean combination of Σ_1^0 accepting condition, and

(2) in the game $G(L(\mathcal{M}))$, Player II has a winning strategy but no computable winning strategies in this game.

To show this theorem, we need to recall 2-register machines.

A **2-register machine** (2RM) $\mathcal{R}$ is a list of programs

$$\mathcal{R} =< (0 : I_0), (1 : I_1), ..., (k : I_k) >$$

where the first column denotes the line number and the second is the instruction. *Instructions* include:

- INC(X_i): increase the content of register i by 1,
- DEC(X_i): decrease the content of register i by 1,
- HALT, and
- IF $X_i = 0$, GOTO s, ELSE GOTO s', where $i \in \{0, 1\}$ and $0 \leq s, s' \leq k$.

A **configuration** a 2-register machine is (ℓ, m, n), where ℓ is the line number and m, n represents the contents of the two registers. We say a configuration is a halting configuration if its instruction is HALT. The transitions of a 2-register machines are defined as follows. For $\ell \in \{0, ..., k\}$, $m, n \in \mathbb{N}$, $0 \leq s, s' \leq k$,

$(\ell, m, n) \mapsto_{\mathcal{R}} (\ell + 1, m + 1, n)$, if $I_\ell = \text{INX}(X_0)$,

$(\ell, m, n) \mapsto_{\mathcal{R}} (\ell + 1, m, n + 1)$, if $I_\ell = \text{INX}(X_1)$,

$(\ell, m, n) \mapsto_{\mathcal{R}} (\ell + 1, m - 1, n)$, if $I_\ell = \text{DEC}(X_0)$ and $m \neq 0$,

$(\ell, m, n) \mapsto_{\mathcal{R}} (\ell + 1, m, n)$, if $I_\ell = \text{DEC}(X_0)$ and $m = 0$,

$(\ell, m, n) \mapsto_{\mathcal{R}} (\ell + 1, m, n - 1)$, if $I_\ell = \text{DEC}(X_1)$ and $n \neq 0$,

$(\ell, m, n) \mapsto_{\mathcal{R}} (\ell + 1, m, n)$, if $I_\ell = \text{DEC}(X_1)$ and $n = 0$,

$(\ell, m, n) \mapsto_{\mathcal{R}} (s', m, n)$, if $I_\ell =$ IF $X_0 = 0$, GOTO s, ELSE GOTO s' and $m \neq 0$,

$(\ell, m, n) \mapsto_{\mathcal{R}} (s', m, n)$, if $I_\ell =$ IF $X_1 = 0$, GOTO s, ELSE GOTO s' and $n \neq 0$,

$(\ell, m, n) \mapsto_{\mathcal{R}} (s, m, n)$, if $I_\ell =$ IF $X_0 = 0$, GOTO s, ELSE GOTO s' and $m = 0$,

$(\ell, m, n) \mapsto_{\mathcal{R}} (s, m, n)$, if $I_\ell =$ IF $X_1 = 0$, GOTO s, ELSE GOTO s' and $n = 0$.

We code a configuration (ℓ, m, n) of a 2-register machine as $\ell a^m b^n$. A **run** of a 2-register machine is a sequence of configurations

$$\ell_0 a^{m_0} b^{n_0} \rhd \ell_1 a^{m_1} b^{n_1} \rhd \ell_2 a^{m_2} b^{n_2} \rhd \cdots$$

A run of a 2-register machine is finite if it visits the halting configuration, otherwise it is an infinite sequence of configurations.

We say that $m_0 \in L(\mathcal{R})$ if and only if $\ell_0 a^{m_0} b^{n_0} \rhd \ell_1 a^{m_1} b^{n_1} \rhd \cdots \rhd \ell_s a^{m_s} b^{n_s}$ where $n_0 = 0$ and $\ell_s = \text{HALT}$.

It is known that the halting problem for 2-register machines is undecidable.

Sketch of proof for Theorem 7

Given a 2-register machine $\mathcal{R}$, we construct a game $G_{\mathcal{R}}$ such that" the existence of computable winning strategies for Player II in $G_{\mathcal{R}} \Leftrightarrow$ the decidability of $L(\mathcal{R})$".

We consider the following infinite games. First Player I provides a number m_0 and asks whether $m_0 \in L(\mathcal{R})$ holds or not. Then it is Player II's turn to answer Yes or No.

- If Player II chooses **Yes**, then Player II should provide a sequence of configurations on m_0 of $\mathcal{R}$ to support her argument.
- Otherwise, Player II chooses **No**, then
  - Player I will defend by providing a sequence of configurations on m_0 of $\mathcal{R}$ that he claims correct, and
  - while Player I writing the sequence of configurations, Player II may challenge at the point she believes Player I has cheated.

Then we define the winning condition for Player II in this game.

- In the case that Player II chooses **Yes**. Player II wins by providing a sequence of configurations on m_0 of $\mathcal{R}$ (Σ_1^0-statement).
- In the case that Player II chooses **No**.
  - If Player I defends by providing an infinite sequence of configurations on m_0 of $\mathcal{R}$ and Player II never challenge, then Player I loses (Π_1^0-statement).
  - If Player II challenges somewhere and shows Player I has cheated, then Player II wins (Σ_1^0-statement).

We would like to see more details of how Player II challenge when she answers **No**. Player II provides a modified reverse form of the previous two configurations as illustrated in Figure 9.

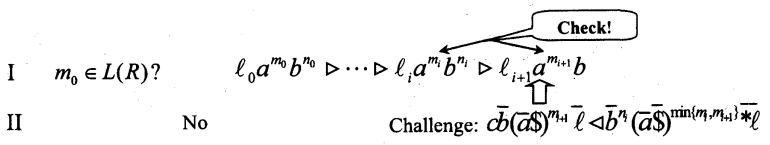

Figure 9 Challenge by Player II.

where $c(\in \mathbf{Int})$ is a letter for challenge, $\$(\in \mathbf{Push}_2)$ and $\bar{\$}(\in \mathbf{Pop}_2)$ are for comparing m_{i+1} with m_i, and $\bar{*}$ is a witness for error.

For this game, the winning set for Player II can be recognized by a 2DVPA with the input alphabet Σ partitioned into

- $\mathbf{Push}_1 = \{a, b, \triangleright, \ell_i\}$ for $\ell_i \in \{I_0, ..., I_k\}$, $\mathbf{Pop}_1 = \{\bar{a}, \bar{b}, \triangleleft, \bar{\ell}\}$,
- $\mathbf{Push}_2 = \{\$\}$, $\mathbf{Pop}_2 = \{\bar{\$}\}$, and
- $\mathbf{Int} = \{c\}$.

We can regard a play in $G_{\mathcal{R}}$ as an input of a 2DVPA as in Figure 10. The operations and conditions of the two stacks while reading the play from left to right are explained as follows.

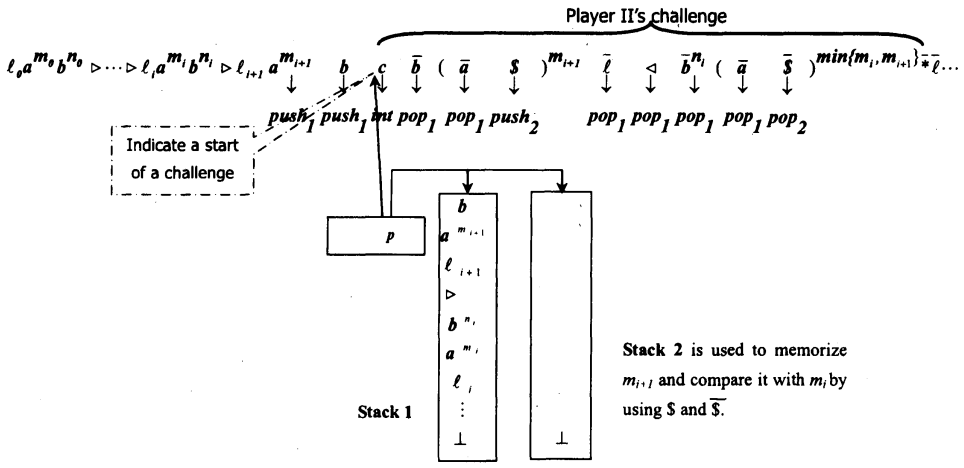

Figure 10 Regarding a play as an input of a 2-stack visibly pushdown automaton.

1 While it reads letters a, b, $\triangleright$, ℓ_i, it pushes the same letters onto the top of Stack 1.

2 Letter c indicates a start of a challenge.

3 By reading $\overline{b}$, it pops the top b in Stack 1.

4 While reading $\overline{a}$ in the section $(\overline{a}\$)^{m_{i+1}}$, it pops one a in Stack 1 and by reading $ it pushes one symbol into Stack 2. In such a way, after reading the section $(a\$)^{m_{i+1}}$, it not only pops all the m_{i+1} many a on the top of Stack 1 but also stores the number of a's it popped from Stack 1 to Stack 2.

5 Letter $\overline{\ell}$ is used to pop the ℓ_{i+1} on the top of the current Stack 1.

6 When reading letter $\triangleleft$, the top element $\triangleright$ in Stack 1 will be popped.

7 By reading n_i many $\overline{b}$, all the b in b^{n_i} on the top of Stack 1 are popped.

8 After reading $(a\overline{\$})^{\min\{m_i,m_{i+1}\}}$, it popped $\min\{m_i, m_{i+1}\}$ many a from the top of Stack 1 and also $\min\{m_i, m_{i+1}\}$ many $ from the top of Stack 2. Then, the current condition of the two stacks might be specified as follows.

   (a) the top of Stack 2 is $\perp$ and that of Stack 1 is a,

   (b) the top of Stack 2 is $ and that of Stack 1 is ℓ_i, or

   (c) other cases.

9 Finally, we meet letter $\overline{*}$, a witness for error.

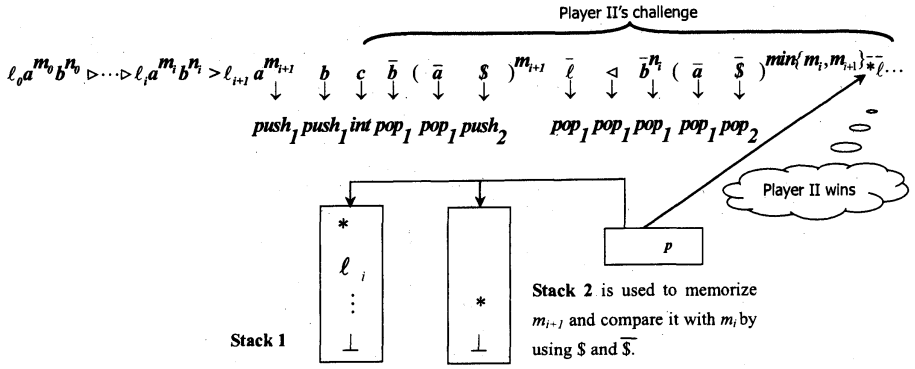

Figure 11 Regarding a play as an input of a 2-stack visibly pushdown automaton.

If satisfying all the following conditions, then Player II succeeds her challenge and wins this game.

i) $\ell_i = INC(X_0) \rightarrow \overline{*} \neq \overline{\$}$, and

ii) $\ell_i = DEX(X_0) \rightarrow \overline{*} \neq \overline{a}$, and

iii) $\ell_i \neq INC(X_0)$ *or* $DEX(X_0) \rightarrow \overline{*} \neq \emptyset$.

For the winning set constructed, Player II clearly has a winning strategy τ such that $L(\mathcal{R}) = \{m_0 |\ \tau(m_0) = \text{"Yes"}\}$. Thus if the halting problem on $\mathcal{R}$ is undecidable, then there is no computable winning strategy for Player II.

□

By sophisticating the above proof, we can also show the following.

Corollary 1. *For any arithmetical set A, there exists a game defined by a 2DVPA with a Boolean combination of Σ_1^0 accepting condition such that Player II has a winning strategy but no simpler than A, i.e., A is computable in any winning strategy for Player II.*

To explain the idea of the proof for Corollary 1, assume that A is a Σ_3^0 subset of $\mathbb{N}$. Then there is a 2-register machine $\mathcal{R}$ such that $m_0 \in A$ if and only if $\exists m_1 \forall m_2 \mathcal{R}$ halts on $m = 2^{m_0}3^{m_1}5^{m_2}$. Now we consider the following game.

- Player I starts the game by asking if $m_0 \in A$.
- Player II answers **Yes** or **No**.
  - If Player II answers **Yes**, she also needs to choose m_1, and then Player I chooses m_2. After that, Player II constructs a sequence of configurations of $\mathcal{R}$ on $m = 2^{m_0}3^{m_1}5^{m_2}$ in the same way as in the proof for Theorem 7.
  - If Player II answers **No**, then the game continues similarly as the roles of the players switched.

The details will appear in future literature. Furthermore, we conjecture that the determinacy of $\mathbf{2DVPL}_\omega$ with $\mathcal{B}(\Sigma_2^0)$ accepting condition is the same as that of the $\mathbf{DTM}_\omega$ (the ω-language recognized by deterministic Turing machines with Muller condition, which is contained in $\mathcal{B}(\Sigma_2^0)$) with respect to the complexity of winning strategies.

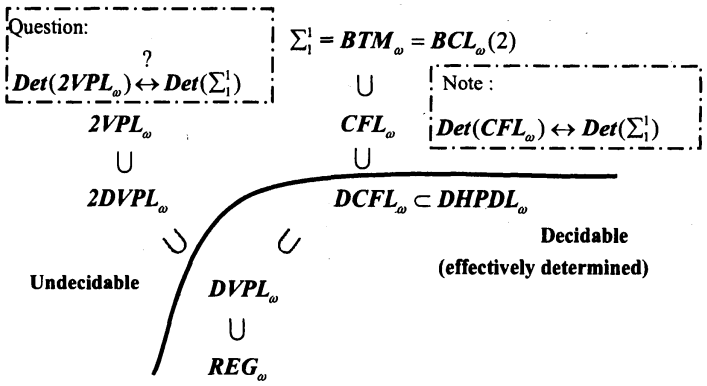

Figure 12 A summary of results on determinacy and computable winning strategy.

4 Conclusions

We have introduced infinite games recognized by 2-stack visible pushdown automata with $\mathcal{B}(\Sigma_1^0)$ accepting condition. We showed that there is such a game with no computable winning strategy. From the perspective of decidability and computable winning strategies in infinite games, our result can be seen as an undecidable counterpart of many studies on games over $\mathbf{DVPL}_\omega$, $\mathbf{DCFL}\omega$ (Walukiewicz, 2001), and $\mathbf{DHPDL}_\omega$ (Cachat, 2003; Carayol, Hague, Meyer, Ong, 2008). We may further extend this study on infinite games to other machines or other accepting conditions. Another interesting problem is to investigate the strength of determinacy of computable games from the viewpoint of reverse mathematics (Nemoto, MedSalem, Tanaka, 2007).

References

[1] R. Alur, and P. Madhusudan, Visibly pushdown languages. In Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing, 2004, 202-211.

[2] J.R. Büchi and L.H. Landweber, Solving sequential conditions by finite-state strategies. Transactions of the American Mathematical Society, 1969(138): 295-311.

[3] T. Cachat, Higher order pushdown automata, the Caucal hierarchy of graphs and parity games. Automata, Languages and Programming. Springer Berlin Heidelberg, 2003, 556-569.

[4] A. Carayol, M. Hague, A. Meyer, C. H. Ong, and O. Serre, Winning regions of higher-order pushdown games. Logic in Computer Science, 2008. LICS'08. 23rd Annual IEEE Symposium on. IEEE, 2008, 193-204.

[5] D. Carotenuto, A. Murano, and A. Peron, 2-visibly pushdown automata. Developments in Language Theory. Springer Berlin Heidelberg, 2007, 132-144.

[6] R.S. Cohen and A.Y. Gold, Theory of ω-languages. Part I: Characterization of ω-context-free languages. Journal of Computer and System Sciences, 1977, 15(2): 169-184.

[7] O. Finkel, Topological complexity of context free ω-languages: A survey Language, Culture, Computation: Studies in Honor of Yaacov Choueka, Lecture Notes in Computer Science, Volume 8001, Springer, 2014, 50-77.

[8] O. Finkel, The determinacy of context-free games. Journal of Symbolic Logic, 2013, 78(4): 1115-1134.

[9] O. Finkel, On omega context free languages which are Borel sets of infinite rank. Theoretical Computer Science, 2003, 299(1): 327-346.

[10] O. Finkel, Topological properties of omega context-free languages. Theoretical Computer Science, 2001, 262(1): 669-697.

[11] W. Fridman, C. Löding and M. Zimmermann, Degrees of lookahead in context-free infinite games. LIPIcs-Leibniz International Proceedings in Informatics. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2011, 12.

[12] C. Löding, Decision problems for deterministic pushdown automata on infinite words. Proceedings 14th International Conference on Automata and Formal Languages, EPTCS 151, 2014, 55-73.

[13] C. Löding, P. Madhusudan and O. Serre, Visibly pushdown games. Foundations of Software Technology and Theoretical Computer Science. Springer Berlin Heidelberg, 2005, 408-420.

[14] T. Nemoto, M. Y. Ould MedSalem, K. Tanaka, Infinite games in the Cantor space and subsystems of second order arithmetic. Mathematical Logic Quarterly, 2007, 53(3): 226-236.

[15] I. Walukiewicz, Pushdown processes: Games and model-checking. Information and Computation, 2001, 164(2): 234-263.